AD-A094 895 CLAREMONT MEN'S COLL CA INST OF DECISION SCIENCE F/G 12/1
COMPUTING LOWER CONFIDENCE BOUNDS WITH THE LIKELIHOOD RATIO MET--ETC(U)
JAN 81 J B LUCKE N00014-78-C-0213
UNCLASSIFIED 81-2 NL

1 OF 1
AD
A094895

END
DATE
FILMED
3-81
DTIC

INSTITUTE OF DECISION SCIENCE
FOR BUSINESS & PUBLIC POLICY

COMPUTING LOWER CONFIDENCE
BOUNDS WITH THE LIKELIHOOD RATIO METHOD

by

James B. Lucke

Report #81-2

January 1981

Research supported in part
by Office of Naval Research
Contract N00014-78-C-0213

Claremont Men's College
Claremont, California

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: 81-2
2. GOVT ACCESSION NO.: AD-A094 895
3. RECIPIENT'S CATALOG NUMBER:
4. TITLE (and Subtitle): Computing Lower Confidence Bounds with the Likelihood Ratio Method
5. TYPE OF REPORT & PERIOD COVERED: Technical
6. PERFORMING ORG. REPORT NUMBER: 81-2
7. AUTHOR(s): James B. Lucke
8. CONTRACT OR GRANT NUMBER(s): N00014-78-C-0213
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Institute of Decision Science, Claremont Men's College, ~~Claremont, California 91171~~
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: Reliability
11. CONTROLLING OFFICE NAME AND ADDRESS: Office of Naval Research, 800 N. Quincy, Arlington, VA 22217
12. REPORT DATE: January 1981
13. NUMBER OF PAGES: 24
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office):
15. SECURITY CLASS. (of this report): Unclassified
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:
16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE: DISTRIBUTION UNLIMITED.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

N/A

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Lower confidence bounds
Systems Reliability
Fixed points

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Subroutines for computing lower confidence bounds are described that use the properties of the method to compute efficiently and avoid possible pitfalls of computing fixed points.

INTRODUCTION

When expensive, highly reliable systems are produced for a customer, the customer often demands more assurance than a point estimate of the reliability from component by component test data. Many methods [IDS 1979] have been proposed for computing lower confidence bounds for series systems but few methods have been extended to general coherent systems including pseudo-series [IDS 1980], a structure which may be used to model certain types of missiles.

The likelihood ratio method, introduced by Madansky [M] and extended to all coherent systems by Myhre and Saunders [M,S], forms the basis of many methods that have proven useful for series systems [IDS 1979] and pseudo-series [IDS 1980]. The computations of this method are difficult to do by hand for series systems and can be so complicated for other systems as to necessitate a computer program. Subroutines which do these computations for pseudo-series and other types of coherent systems are the focus of this report.

THE LIKELIHOOD RATIO METHOD

In [M], Madansky introduced the likelihood ratio method of computing confidence intervals for series and parallel systems with binomial component data; this method is based on a result of Wilks that $-2\ln L(r)$ is distributed asympotically as a chi-squared with one degree of freedom, where $L(r)$ is the likelihood ratio:

$$L(r) = \frac{\sup\ B(x_i,n_i;p_i)}{\sup\ B(x_i,n_i;p_i)}\quad ,$$

where $B(x_i,n_i;p_i)$ is the binomial distribution and where the sup in the numerator is over all p such that $r=\Pi\, p_i$ while the sup in the denominator is over all possible values of p and is achieved when p is the maximum likelihood estimate $p_i=x_i/n_i$. In [M] Madansky shows that the numerator can be made a function of δ, a Lagrange multiplier, with $p_i=(x_i-\delta)/(n_i-\delta)$ and $r=\Pi\, p_i$. Hence it is straightforward to solve for $\delta_1<0$ and $\delta_2>0$ by finding the two solutions of $\ln L(p(\delta))=-\chi^2(1)/2$, where $\chi^2(1)$ is the $1-\gamma$ percentile of the chi-sqaured distribution with one degree of freedom. The resulting confidence interval is $S(r)=\{r: h(p(\delta_2))<r<h(p(\delta_1))\}$, where $h(p)$ is the reliability function of the system evaluated at the vector p and $p_i(\delta)=(x_i-\delta)/(n_i-\delta)$.

In [M,S] Myhre and Saunders extended this method to general coherent systems with the resulting computations being more complicated than those in the series case. Using the method of Lagrange with multiplier δ to find the sup of the numerator results in the equations:

$$h(p)=r$$

$$D_j\ \{\ \Sigma \ln B(x_i,n_i;p_i) - h(p)\} = 0\ ,\quad 1<j<m$$ where $h(p)$ is the reliability function of the coherent system and $p=(p_1,p_2, \ ,p_m)$ is the vector of component reliabilities. Myhre and

Saunders show that under certain conditions p is a decreasing function of δ (δ>0) and exists as a fixed point of a contractive map. Representing this fixed point as p(δ), $\ln(L(p))=-\chi^2(1)/2$ may be turned into an equation in δ. As $\ln\{L(p(\delta))\}$ is decreasing in δ, we can bracket $-\chi^2(1)/2$ by incrementing from zero. Then using a technique like the secant method to solve for δ we may find the confidence interval [R,1] , where R=h(p(δ)) .

In practice , it turns out that the contractive map may be sensitive to changes in δ near the solution and also may be erratic when the map is started far from the fixed point. The present subroutines used to calculate the confidence intervals are written to extrapolate (or interpolate) to a guess at the new fixed point each time a new δ is used. By doing this the programs have run efficiently and trouble-free.

The Subroutines for General Coherent Systems

There are two subroutine packages, one for general coherent systems and another for pseudo-series systems only. Each package has three subroutines, LRSUB(PLRSUB), POSDEL,AND CONMAP, as well as two functions, XLAMBDA and RELFUN. THEY are written in fortran 77 and run on the CLAREMONT COLLEGES VAX computer when combined with a driving program that sets up the data. The subroutines in the package for general coherent systems and RELFUN share a common block of data that contains the structure of the system, the test data, and the binomial component test data:

Structure of the System

ORDER is the number of distinct components in the system

ITYPE is a parameter of the type of system

1:series

2:parallel

3:k out of n

Binomial Test Data

WINS(I) is the vector of number of successful trials for each component

TRIALS(I) is the vector of number of times each component was tested.

Common Computational Variables

DELTA is the lagrange multiplier

OLDLAM is the value of the function $\Lambda(p_i)$ in [M,S] for the present delta.

PDELTA(I) is the vector of $p_i(\delta)$'s.

PTILDE(I) is the vector of maximum likelihood

estimates of the p_i

LRSUB sets up the variables and calls POSDEL three times to compute 80,85,and 90% lower confidence bounds.POSDEL solves the equation $(p(\ ))=-\chi^2(1)/2$ by incrementing δ until the solution is bracketed and then interpolating to the solution. POSDEL calls CONMAP to get $p(\delta)$ for each δ and uses XLAMBDA to compute $\Lambda(p(\delta))$ once $p(\delta)$ is known. When the equation is solved, control returns to LRSUB from POSDEL and LRSUB computes the bound by plugging the solution vector $p(\delta)$ into RELFUN , the reliability function of the system.

LRSUB

LRSUB is the subroutine that drives the programs that compute 80,85,and 90% lower confidence bounds for general coherent systems using the likelihood ratio method. LRSUB sets up CHISQ(3) as $\chi^2(1)$ for the three levels; sets up p=.1 min{ WINS(I): 1<I< ORDER } to be used as an increment of DELTA in POSDEL; and initializes PDELTA(I)= PTILDE(I)= WINS(I)/ TRIALS(I). Then LRSUB calls the subroutine POSDEL once for each of the three confidence levels with the appropriate $Q=-\chi^2(1)/2$; computes the bound by using RELFUN and the returned PDELTA; and stores the computed bounds in LCB(I),I=1,3. LRSUB calls POSDEL , while POSDEL calls CONMAP and XLAMBDA.

POSDEL

POSDEL recieves the arguments (P,Q) and uses DELTA and the other variables in the common block to solve $(p(\delta))$=XLAMBDA(PDELTA)=Q= $-\chi^2(1)/2$ as an equation in δ=DELTA, where PDELTA is a vector of length ORDER determined by CONMAP for each DELTA . POSDEL takes into account the sensitivity of the contractive map in CONMAP and the need for computational efficiancy by extrapolating (or interpolating) from the previous values of PDELTA to get a new PDELTA which corresponds to the new DELTA which is sent to CONMAP.

To keep the solution process clear , POSDEL is written in two parts: the first part to bracket Q by finding successive DELTA such that XLAMBDA(DELTA-GAP)>Q>XLAMBDA(DELTA) ; and the second part uses the secant method to find a DELTA such that !XLAMBDA(DELTA) - Q!<.001. To keep track of the past results, enabling us to extrapolate and interpolate for PDELTA , LWBD(I), and UPBD(I), I=1,ORDER are set up to contain the PDELTAs that gave the closest upper and lower bounds on Q (LWBD(I) is initialized as 0.0 and UPBD(I) as x_i/n_i for the first bound and as the solution to the last bound for the second and third bounds).

In part one DELTA is incremented by the value of GAP , which is initially the value of P ,sent by LRSUB, and for each new DELTA the vector of PDELTA's is found by CONMAP and OLDLAM is computed by XLAMBDA. Part one iterates through values of DELTA until Q>OLDLAM ([M,S] shows that XLAMBDA is a decreasing function of DELTA). Once we have PDELTA for one DELTA , we extrapolate for a guess at the next PDELTA so that it corresponds to the new DELTA =DELTA +GAP ; let $p_I=P(\delta_1)$, $p_2=P(\delta_2)=P(\delta+\Delta)$ and assume $P(\delta)$ is linear with $p_3=P(\delta_2+\Delta)$. Then $p_3-p_2=\text{slope}*\Delta=(p_2-p_1)/\Delta*\Delta$ and $p_3=2p_2-p_1$. Thus the PDELTA sent

to CONMAP is given as

PDELTA(I)=2*UPBD(I)-OLP(I) , 1<I<ORDER where OLP is the PDELTA from the previous step.

If CONMAP fails when called in part one , GAP is halved so that POSDEL will approach the solution region more slowly. An analysis similiar to the above shows that the extrapolated PDELTA should then be:

PDELTA(I)=3/2 UPBD(I) -1/2 OLP(I) , 1<I<ORDER. To keep LWBD and UPBD correct they are updated (after saving UPBD in OLP) before completing the loop. Until Q is bracketed the new PDELTA returned by CONMAP is saved in UPBD while PDELTA is stored in LWBD when OLDLAM<Q and the program moves to part two.

Upon entering part two , Q is bracketed and hence we may estimate the solution by the interpolation:

$\delta = RR(\delta_2 - \delta_1) +$ where $RR = (Q - \lambda_1)/(\lambda_2 - \lambda_1)$. Using the same interpolation $p(\delta) = RR(p_2 - p_1) + p_1$ or PDELTA(JJ)= UPBD(JJ)+RR*(LWBD(JJ)-UPBD(JJ) , 1<JJ<ORDER. Once DELTA and PDELTA are set up , CONMAP is called and if it does not fail LWBD and UPBD are updated depending on whether the new OLDLAM is less than Q or larger than Q. If CONMAP fails in part two , RR is halved and CONMAP tried again until CONMAP works or has failed nine times (CONMAP works most easily if it does not have to find PDELTA corresponding to a large change in DELTA). DELSAV and DL are also updated before a new interpolation is completed. The loop in part two is thus implementing the secant method and continues until !Q-OLDLAM!<.001 at which time control returns to LRSUB where the bound is computed.

CONMAP

CONMAP receives as arguments FAIL,UPBD,LWBD, and GAP, and uses DELTA and the other variables in the common block to compute the fixed point of the contractive map $A((p_i),\delta)$ of Theorem 2-4 of [M,S]: if $(q_i)=A((p_i),\delta)$, and if $a_j(p)=p_j*D_jh(p)$ then $q_j=(x_j-a_j(p))/(n_j-a_j(p))$. In [B,P] it is shown that $D_jh(p)=h(p_0)-h(p_1)$, where p_1 is the vector of p_i's with $p_j=1$ and p_0 is the same vector with $p_j=0$. In the program AJP=$A_j(p)$ for each j and q_j=A(J)=(WINS(J)-DELTA*AJP)/(TRIALS(J)-DELTA*AJP). This is not possible if WINS(J) is less than DELTA*AJP and in that case A(J) is left at the previous value (which is PDELTA(J) during the first time through CONMAP). If the new value of A(J) is not in the interval [UPBD(J),LWBD(J)] it is set equal to (PDELTA(J)+BD)/2 where BD is the bound it exceeded.

As the new A(J) are computed , DIST is updated so that it is the distance between the new and old vector values of the contractive map. The map fails if this distance doesn't contract at each step , which is the result if DIST>DIST2. Otherwise DIST2 is updated and the process continues until we find a fixed point (DISTTOL=.0005) and CONMAP returns successfully. HOLD(I) is used to save the initial value of PDELTA to be returned if the contractive map fails. If GAP is small (GAP<.00001), PDELTA is determined by averaging UPBD and LWBD since little change can be expected from the contractive map in such a case.

RELFUN

RELFUN is a subfunction that computes the reliability of the given type of system. ITYPE determines the type of system: ITYPE = 1,2, or 3 if the system is respectively series,parallel , or k out of n system. This function takes a vector of probabilities, PROB , as its argument and computes the reliability using the appropriate formula:

ITYPE=1: series system; $r = \Pi prob(i)$

ITYPE=2: parallel system: $r = 1 - \Pi(1-prob(i))$

ITYPE=3: k out of n system : $r = \Sigma E(i;n;p)$,

where $E(i;n;p)$ is the probability of exactly i successes out of n

```
ty
C***************** THIS IS THE SUBROUTINE LRSUB ******************
C       IT RUNS THE LIKELIHOOD RATIO METHOD
C       IT DOES NOT ADJUST THE FAILURES
C       IT WORKS FOR ALL SYSTEMS
C       WHOSE RELIABILITY FUNCTION CAN BE WRITTEN IN RELFUN
C               NOW ON VAX. 2/28/80
C               ALL IS WORKING WELL WITH THE
C               SUBROUTINES POSDEL,CONMAP
C*****************************************************************

        SUBROUTINE LRSUB(LCB)
        DIMENSION CHIISQ(3)
        INCLUDE 'lrcomr.for'
        INTEGER ORDER
        REAL LCB(3)
        DATA CHIISQ/.709, 1.074,  1.642/
C****   P GIVES US AN INITIAL INCREMENT BY WHICH WE CHANGE
C****   DELTA TO LOCALIZE ITS VALUE
        P=1.E15
           DO 40 I=1,ORDER
           IF(WINS(I).LE.P)P=WINS(I)
           PTILDE(I)=WINS(I)/TRIALS(I)
           PDELTA(I)=PTILDE(I)
40         CONTINUE
        P=P/10.0
                DELTA=P
                OLDLAM=0.0
           DO 50 I=1, 3
           Q=-1.0*CHIISQ(I)/2
           CALL POSDEL(BOUND,P,Q)
           BOUND=RELFUN(PDELTA)
           LCB(I)=BOUND
50         CONTINUE
        write(23,1514)(LCB(I),I=1,3)
1514    format(2x,'The Bounds are :',3(f8.5,2x))
        RETURN
999     END
C************************ END OF LRSUB ***********************
C
C
C
C
C     SUBROUTINE POSDEL ITERATES THROUGH VALUES FOR DELTA PLUS
C
C************************* POSDEL ***************************
C       POSDEL HAS TWO MAIN parts: in part 1 the value
C       of delta is incremented, with conmap finding
c       the vector of p(i)s,until q has been bracketed.
c       Once a vector of p(i)s is known we extrapolate
C       to get a new vector to start conmap with(the vector
C       corresponds to the new value of delta). This makes
C       conmap less likely to fail. If conmap fails the size
C       of the increment for delta is halved.
C       Part 2 interpolates to get a new delta and also
C       interpolates the p(i)s to get a corresponding vector
C        to start conmap. The interpolation is made possible
C       by storing the p(i)s that correspond to the upper
C       and lower bounds on Q.
C                         JBL   3/6/80
C ***********************************************************

        SUBROUTINE POSDEL(BOUND,P,Q)
```

```
      DIMENSION UPBD(100),OLP(100)
      REAL LWBD(100)
      INTEGER ORDER
      LOGICAL FAIL
C******************** INITIALIZING **************************
      GAP=P
         DO 121 I=1,ORDER
         LWBD(I)=0.0
             UPBD(I)=PDELTA(I)
121      CONTINUE
      FAIL=.FALSE.
C******************** PART 1 *************************
C START OF THE SEARCH FOR A SOLUTION REGION BY
C  INCREMENTING DELTA BY GAP UNTIL THE SOLUTION IS BRACKETED
C
      DO 20 J=1,200
      OLAMB=OLDLAM
      DELSAV=DELTA
      DELTA=DELTA+GAP
             IF(J.GT.1) THEN
C            ****** here we extrapolate for pdelta *******
             DO 1042 JJ=1,ORDER
             PDELTA(JJ)=2*UPBD(JJ)-OLP(JJ)
             IF(FAIL)PDELTA(JJ)=1.5*UPBD(JJ)-.5*OLP(JJ)
1042         CONTINUE
             END IF
      CALL CONMAP(FAIL,UPBD,LWBD,GAP)
1044         IF(FAIL) THEN
c        ------ block 1-------------------------------------
              TYPE 9870
9870         format('  conmap failed ')
             DELTA=DELTA-GAP
             GAP=GAP/2.0
C        ----  block 2 -----------------------------------
             ELSE
             OLDLAM=XLAMBD(TRIALS,PTILDE,PDELTA,ORDER)
d     rel=relfun(pdelta)
d     write(5,1111) delta,oldlam,q,rel
d     write(5,1112) (pdelta(ITST),ITST=1,order)

1111  format(2x,4f10.4,'##',/,8f6.4)
1112  format(f8.4)

c   ++++++++++++++++++++++++++++++++++++++++++++
                IF(ABS(OLDLAM-Q).LT..001)   RETURN
c  +++++++++++++++++++++++++++++++++++++++++++++++
                DO 19 JJ=1,ORDER
                OLP(JJ)=UPBD(JJ)
                IF(OLDLAM.LT.Q) LWBD(JJ)=PDELTA(JJ)
                IF(OLDLAM.GT.Q) UPBD(JJ)=PDELTA(JJ)
19              CONTINUE
                IF(OLDLAM.GT.Q)OLAMBU=OLDLAM
c               ------------------------------------
C          **** THIS IS OUR GOAL FOR THIS PART
         IF(OLDLAM.LT.Q) GOTO 30
c --------------- end of block 2 -------------------------
             END IF
20       CONTINUE
c ---------------------------------------------------------
      WRITE(5,2)
2     format(4x,'No solution after 100 steps in part 1.')
      BOUND=.01
      RETURN
C**************** END OF PART 1 ***********************
C
```

```
C  THIS IS THE START OF THE SECTION THAT SOLVES FOR
C       DELTA USING THE REGULA FALSI METHOD
C       AND we interpolate for the pdelta (between upbd
C       and lwbd.).
C************************************************************

c     **** INITIALIZE THE VARIABLES ******************
30      deltau=delta-gap
        deltal=delta
        olambl=oldlam
c       -----------------------------------------------------
        DO 40 J=1,100
c       **** here we interpolate for delta and pdelta *****
        IF(.NOT.FAIL) RR=(Q-OLAMBU)/(OLAMBL-OLAMBU)
        IF(FAIL) RR=RR/2.0
        DELTA=DELTAU+RR*(DELTAL-DELTAU)
                DO 1313 JJ=1,ORDER
                PDELTA(JJ)=UPBD(JJ)+RR*(LWBD(JJ)-UPBD(JJ))
1313            CONTINUE
        GAPP=DELTAL-DELTAU
c       -----------------------------------------------------
        CALL CONMAP(FAIL,UPBD,LWBD,GAPP)
                IF(.NOT.FAIL) THEN
c               ----- block 1 -------------------------------
                IT=0
9871            OLDLAM=XLAMBD(TRIALS,PTILDE,PDELTA,ORDER)
c   ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        IF((OLDLAM-Q).lt..001)RETURN
c   ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                IF(oldlam.lt.Q) THEN
c               -- block a------------------
                DELTAL=DELTA
                OLAMBL=OLDLAM
                do 38 jj=1,order
                LWBD(JJ)=PDELTA(JJ)
38              CONTINUE
c               ------------------
                ELSE
c               --- block b -----------
                DELTAU=DELTA
                OLAMBU=OLDLAM
                DO 39 JJ=1,ORDER
                UPBD(JJ)=PDELTA(JJ)
39              CONTINUE
c               ------------------------
                END IF
C           ****   CHECK TO SEE IF DELTAS ARE TOO CLOSE *****
                IF((deltal-deltau).lt.1E-5) then
                WRITE(5,3)
3               FORMAT(' DELTA CAN NO LONGER BE signif. INCREASED.')
                RETURN
                END IF
c             ---------------------------------------------
        ELSE
c       ------CRISIS ROUTE ------------------------------------------
                IT=IT+1
                IF(IT.gt.9)stop
        END IF
c       -----------------------------------------------
40         CONTINUE
        WRITE(5,1)
1       FORMAT(' NO SOLUTION FOUND AFTER 100 times thru part 2')
        BOUND=.01
        RETURN
c       ----------- unsuccessful exit ------------------------------
```

```
c
c       THIS COMPUTES THE CONTRACTIVE MAP
C       OF EQN. 2.1 IN MYHRE-SAUNDERS PAPER.
C               UPDATED TO JUST KEEP A(J) BETWEEN THE
C               BOUNDS ON PDELTA.      JBL    2/16/80
C               THIS VERSION IS FOR GENERAL COHERENT
C               SYSTEMS AND IS USED BY LRSUB. JBL 5/22/80
C************************  CONMAP  *****************************

        SUBROUTINE CONMAP(FAIL,UPBD,LWBD,GAP)
        INCLUDE 'lrcomr.FOR'
        INTEGER ORDER
        LOGICAL FAIL
        DIMENSION HOLD(100),A(100),UPBD(100)
        REAL LWBD(100)
        FAIL=.FALSE.
        TOL=.0005
C       ****  IF GAP IS VERY SMALL WE SIMPLY AVERAGE THE BOUNDS OF PDELTA.
        IF(GAP.LT.1E-4)THEN
        DO 999 J=1,ORDER
        PDELTA(J)=(LWBD(J)+UPBD(J))/2.0
999     CONTINUE
        RETURN
c       ------------------------------
        ELSE
C       MAIN PART OF WORK.
        DO 10 J=1,ORDER
        HOLD(J)=PDELTA(J)
        A(J)=HOLD(J)
10      CONTINUE
C*****  SOLVE THE CONTRACTIVE MAP
        DIST2=1.0
100     DO 40 ICE=1,1000
        DIST=0.
c               --- compute delaj for each component -------
                DO 30 J=1,ORDER
                IF(WINS(J).EQ.TRIALS(J))A(J)=1.
                IF (A(J).NE.1.0) THEN
                temp=pdelta(j)
                pdelta(j)=0.0
                p0=relfun(pdelta)
                pdelta(j)=1.0
                p1=relfun(pdelta)
                pdelta(j)=temp
                delaj=delta*pdelta(j)*(p1-p0)
C                 *** IF DELAJ>=WINS, DO NOT ASSIGN A(J), IT WILL BE NEG.
                        IF(DELAJ.LT.WINS(J)) THEN
                        A(J)=(WINS(J)-DELAJ)/(TRIALS(J)-DELAJ)
C                       *** WE MUST MAKE SURE A(J) STAYS BETWEEN THE BOUNDS
                        IF(A(J).LT.LWBD(J))A(J)=  (PDELTA(J)+LWBD(J))/2.0
                        IF(A(J).GT.UPBD(J))A(J)=(PDELTA(J)+UPBD(J))/2.0
                        DIST=DIST+ABS(A(J)-PDELTA(J))
c                       ---------------------------------------------
                        END IF
                END IF
30              CONTINUE
c       -------------------- new pdeltas are done ----------------------
                DO 70 L=1,ORDER
                PDELTA(L)=A(L)
70              CONTINUE
c       *****   IF APPROXIMATIONS HAVE CONVERGED ENOUGH, WE STOP ITERATING.
           DIST=DIST/ORDER
```

```
C       ++++++++++++++++++++++++++++++++++++++++++++++++++++++
           IF(DIST.LT.TOL) RETURN
C       ++++++++++++++++++++++++++++++++++++++++++++++++++++++
C       *****   IF DIST IS INCREASING, CONMAP FAILS AND WE GO TRY
C       *****   ANOTHER DELTA AND KEEP THE OLD PDELTA'S.
           IF(DIST.LT.DIST2) THEN
41              DIST2=DIST
C               ------------------------
                ELSE
                DO 50 J=1,ORDER
50              PDELTA(J)=HOLD(J)
                FAIL=.TRUE.
                RETURN
C               ------------------------
                END IF
40      CONTINUE
        END IF
        END
C*****************************************************
$
```

Subroutines for Pseudo-series Systems

The subroutine package for pseudo-series systems consists of three subroutines, PLRSUB,POSDEL, and CONMAP as well as two functions, XLAMBDA and RELFUN. These subroutines perform the same tasks as those in the other package , but are specialized for pseudo-series since the reliability function of a psuedo-series is given as $h(p)=\Sigma w_j \Pi p^{\alpha_{ij}}$ where $\alpha_{ij}>0$ if the i-th component is in the j-th subpopulation and $\alpha_{ij}=0$ if not. See [IDS 1980] for a derivation of this function that shows how it models some missile systems. From this form of the reliability function it follows that $D_k h(p) * p_k = \Sigma w_j \alpha \Pi p^{\alpha}$. In this package the part of the common block describing the structure of the system is expanded so that it contains:

ORDER is the number of distinct components in the system.

LLL is the number of subpopulations in the system.

W(J) is the vector of subpopulation weights.

AL(I) is the vector of component exponents.

IG(I,J) is the matrix of indices for the inclusion of component j in the i-th subpopulation. The rest of the common block is the same as in the other package.

PLRSUB performs exactly the same role in this package as LRSUB does in the other package. POSDEL is the same as the other POSDEL, while CONMAP is almost the same except for the lines computing AJP. Since $a_k(p)=p_k * D_k h(p)= \Sigma w_j \alpha \Pi p^{\alpha}$ and is not $p(h(p_0)-h(p_1))$, AJP is computed directly in this version of CONMAP. XLMBDA is the same as before, but of course RELFUN is the implimentation of the pseudo-series reliability with no possibility of modelling anything else.

```
ty pjustlr.for
C**************    THIS IS PJUSTLR ***************
C
C                           REVISED 2/8/80    JBL
C****************************************************
        REAL LCB(3)
        INTEGER ORDER
        INCLUDE 'lrcomp.FOR'
        CALL PGETDAT(WINS)
        CALL PLRSUB(LCB)
999     END
C     ************************************************
C
C
C
C***************** SUBROUTINE PGETDAT ****************
C                    *******************
C       THIS WILL SET UP THE DATA FOR COMPUTING LIKELIHOOD
C       RATIO LOWER CONFIDENCE BOUNDS FOR PSEUDO-SERIES BY
C       ASKING FOR THE INPUT AND OUTPUT FILES AND OPENING THE FILES.
C       THE INPUT FILE MUST CONTAIN IN ORDER:
C               NUMBER OF UNIQUE COMPONENTS (N) AND SUBSYSTEMS (L)
C               WEIGHTS FOR THE L SUBSYSTEMS
C               EXPONENTS FOR EACH UNIQUE COMPONENT
C               NXL MATRIX INDICATING WHICH OF THE
C                 L SUBSYSTEMS EACH COMPONENT IS
C                 COMMON TO.
C               SUCCESSES AND TRIALS FOR COMPONENTS
C                 1,2,...,N
C
C*******************************************************************
        SUBROUTINE PGETDAT(OWINS)
        DIMENSION OWINS(100)
        INTEGER ORDER
        DOUBLE PRECISION FILNAM
        INCLUDE 'lrcomp.FOR'
15      FORMAT(2i2)
        TYPE 1313
1313    FORMAT(' WHAT FILE HAS THE SYS. DATA ',$)
        accept 1314, FILNAM
1314    FORMAT(A10)
        OPEN(UNIT=22,name=filnam,ACCESS='sequential',type='old')
        TYPE 1315
1315    FORMAT('  WHAT FILE SHOULD THE OUTPUT
     2 BE FILED IN?'$)
        accept 1314,FILNAM
       open(unit=23,name=filnam,access='sequential',type='new')
        READ(22,15)ORDER,LLL
        IF (ORDER .EQ. 0) GOTO 30
        READ(22,413)(W(I),I=1,LLL)
413     FORMAT(12F)
        write(23,1511)
1511    format(2x,'THE FOLLOWING IS RUN ON THE VAX,',/
     1  ,2x,'using JBEPOS,JBCONM .')
        call IDATE(ID1,id2,id3)
        write(23,1512) id1,id2,id3
1512    format(2x,'The date is ',i3,':',i3,':',i3)
        write(23,1513)ORDER,LLL
1513    format(2x,/,15x,'THE SYSTEM DATA IS :',/,
     1  4x,'THE order is ',i3,' the number of sub is ',i3)
        WRITE(23,512)
512     FORMAT(/,'  THE WEIGHTS ARE: ')
        WRITE(23,413)(W(J),J=1,LLL)
        READ(22,414)(AL(J),J=1,ORDER)
```

```
414     FORMAT(10F)
        WRITE(23,515)
515     FORMAT(/,' THE INDICES                                    THE EXPONENTS ')
           DO 119 J=1,ORDER
           READ(22,415)(IG(I,J),I=1,LLL)
           WRITE(23,416)J,(IG(I,J),I=1,12),AL(J)
415        FORMAT(13I)
416        FORMAT(13I3,F10.6)
119        CONTINUE
           DO 30 I=1,ORDER
           READ(22,125) OWINS(I),TRIALS(I)
30         CONTINUE
125     FORMAT(2F)
        WRITE(23,1225) (OWINS(I),I=1,ORDER)
1225    FORMAT(' WINS:    ',15F7.3)
        WRITE(23,1226) (TRIALS(I),I=1,ORDER)
1226    FORMAT(' TRIALS: '15F7.1)
        RETURN
        END
C*********************** XLAMBDA ********************************
C                 THIS SOLVES EQUATION 1.4
C                 TO FIND LAMBDA(DELTA)
C
C****************************************************************
            FUNCTION XLAMBD(TRIALS,PTILDE,PDELTA,I)
            DIMENSION TRIALS(100),PTILDE(100),PDELTA(100)
            S=0.
            DO 10 J=1,I
        IF(PTILDE(J).EQ.1.0)GOTO 10
            ZLOG1=ALOG(PDELTA(J)/PTILDE(J))
            ZLOG2=ALOG((1.-PDELTA(J))/(1.-PTILDE(J)))
            S=S+TRIALS(J)*(PTILDE(J)*ZLOG1+(1.-PTILDE(J))*ZLOG2)
10          CONTINUE
            XLAMBD=S
            RETURN
            END
C***************************************************************
C
C
C
C
C********************** RELFUN *********************************
C                 THIS IS THE REL. FUNCTION FOR PSEUDO-SERIES
C
C
            FUNCTION RELFUN(PROB)
        INCLUDE 'lrcomp.FOR'
            DIMENSION PROB(100)
        INTEGER ORDER
        S1=0.0
        DO 100 L=1,LLL
        P1=1.0
        DO 90 J=1,ORDER
        P1=P1*PROB(J)**(AL(J)*IG(L,J))
90      CONTINUE
        S1=S1+W(L)*P1
100     CONTINUE
        RELFUN=S1
        RETURN
            END
```

```
ty plrsub.for
C**************** THIS IS THE SUBROUTINE PLRSUB *****************
C       IT RUNS THE LIKELIHOOD RATIO METHOD
C       IT DOES NOT ADJUST THE FAILURES
C       IT WORKS ONLY FOR PSEUDO-SERIES.
C               ALL IS WORKING WELL WITH THE
C               SUBROUTINES POSDEL,CONMAP
C               WORKING ON THE VAX, 5/9/80
C*****************************************************************

        SUBROUTINE PLRSUB(LCB)
        DIMENSION CHIISQ(3)
        INCLUDE 'lrcomp.for'
        INTEGER ORDER
        REAL LCB(3)
        DATA CHIISQ/.709, 1.074,  1.642/
C****   P GIVES US AN INITIAL INCREMENT BY WHICH WE CHANGE
C****   DELTA TO LOCALIZE ITS VALUE
        P=1.E15
           DO 40 I=1,ORDER
           IF(WINS(I).LE.P)P=WINS(I)
           PTILDE(I)=WINS(I)/TRIALS(I)
           PDELTA(I)=PTILDE(I)
40         CONTINUE
        P=P/10.0
                DELTA=P
                OLDLAM=0.0
           DO 50 I=1, 3
           Q=-1.0*CHIISQ(I)/2
           CALL POSDEL(BOUND,P,Q)
           BOUND=RELFUN(PDELTA)
           LCB(I)=BOUND
50         CONTINUE
        RETURN
999     END
C************************ END OF PLRSUB ************************
C
C
C
C
C************************  POSDEL  **************************
C       POSDEL HAS TWO MAIN parts: in part 1 the value
C       of delta is incremented, with conmap finding
c       the vector of p(i)s,until q has been bracketed.
c       Once a vector of p(i)s is known we extrapolate
C       to get a new vector to start conmap with(the vector
C       corresponds to the new value of delta). This makes
C       conmap less likely to fail. If conmap fails the size
C       of the increment for delta is halved.
C       Part 2 interpolates to get a new delta and also
C       interpolates the p(i)s to get a corresponding vector
C        to start conmap. The interpolation is made possible
C       by storing the p(i)s that correspond to the upper
C       and lower bounds on Q.
C                       JBL   3/6/80
C               NOW WORKING ON VAX  2/28/80
C   ***********************************************************

        SUBROUTINE POSDEL(BOUND,P,Q)
        INCLUDE 'lrcomp.FOR'
        DIMENSION UPBD(100),OLP(100)
        REAL LWBD(100)
        INTEGER ORDER
        LOGICAL FAIL
```

```
C******************** INITIALIZING ***************************
       GAP=P
          DO 121 I=1,ORDER
          LWBD(I)=0.0
               UPBD(I)=PDELTA(I)
121       CONTINUE
       FAIL=.FALSE.
C******************** PART 1 ****************************
C START OF THE SEARCH FOR A SOLUTION REGION BY
C  INCREMENTING DELTA BY GAP UNTIL THE SOLUTION IS BRACKETED
C
       DO 20 J=1,200
       OLAMB=OLDLAM
       DELSAV=DELTA
       DELTA=DELTA+GAP
               IF(J.GT.1) THEN
C              ****** here we extrapolate for pdelta *******
               DO 1042 JJ=1,ORDER
               PDELTA(JJ)=2*UPBD(JJ)-OLP(JJ)
               IF(FAIL)PDELTA(JJ)=1.5*UPBD(JJ)-.5*OLP(JJ)
1042           CONTINUE
               END IF
       CALL CONMAP(FAIL,UPBD,LWBD,GAP)
1044           IF(FAIL) THEN
c         ------ block 1---------------------------------------
                TYPE 9870
9870           format('  conmap failed ')
               DELTA=DELTA-GAP
               GAP=GAP/2.0
C         ----  block 2 ------------------------------------
               ELSE
               OLDLAM=XLAMBD(TRIALS,PTILDE,PDELTA,ORDER)
c   ++++++++++++++++++++++++++++++++++++++++++++++++
                  IF(ABS(OLDLAM-Q).LT..001)   RETURN
c  +++++++++++++++++++++++++++++++++++++++++++++++++++
                  DO 19 JJ=1,ORDER
                  OLP(JJ)=UPBD(JJ)
                  IF(OLDLAM.LT.Q) LWBD(JJ)=PDELTA(JJ)
                  IF(OLDLAM.GT.Q) UPBD(JJ)=PDELTA(JJ)
19                CONTINUE
                  IF(OLDLAM.GT.Q)OLAMBU=OLDLAM
c                 ------------------------------------
C           **** THIS IS OUR GOAL FOR THIS PART
          IF(OLDLAM.LT.Q) GOTO 30
c --------------- end of block 2 -------------------------
               END IF
20        CONTINUE
c ---------------------------------------------------------
       WRITE(5,2)
2      format(4x,'No solution after 100 steps in part 1.')
       BOUND=.01
       RETURN
C**************** END OF PART 1 ***********************
C
C****************** PART2 *********************************
C  THIS IS THE START OF THE SECTION THAT SOLVES FOR
C      DELTA USING THE REGULA FALSI METHOD
C      AND we interpolate for the pdelta (between upbd
C      and lwbd.).
C*********************************************************

c     **** INITIALIZE THE VARIABLES *****************
30     deltau=delta-gap
       deltal=delta
       olambl=oldlam
```

```
c       **** here we interpolate for delta and pdelta *****
        IF(.NOT.FAIL) RR=(Q-OLAMBU)/(OLAMBL-OLAMBU)
        IF(FAIL) RR=RR/2.0
        DELTA=DELTAU+RR*(DELTAL-DELTAU)
                DO 1313 JJ=1,ORDER
                PDELTA(JJ)=UPBD(JJ)+RR*(LWBD(JJ)-UPBD(JJ))
1313            CONTINUE
        GAPP=DELTAL-DELTAU
c       -----------------------------------------------------
        CALL CONMAP(FAIL,UPBD,LWBD,GAPP)
                IF(.NOT.FAIL) THEN
c               ----- block 1 -------------------------------
                IT=0
9871            OLDLAM=XLAMBD(TRIALS,PTILDE,PDELTA,ORDER)
c   ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        IF((OLDLAM-Q).lt..001)RETURN
c   ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                IF(oldlam.lt.Q) THEN
c               -- block a-------------------
                DELTAL=DELTA
                OLAMBL=OLDLAM
                do 38 jj=1,order
                LWBD(JJ)=PDELTA(JJ)
38              CONTINUE
c               -------------------
                ELSE
c               --- block b -----------
                DELTAU=DELTA
                OLAMBU=OLDLAM
                DO 39 JJ=1,ORDER
                UPBD(JJ)=PDELTA(JJ)
39              CONTINUE
c               -----------------------
                END IF
C           ****   CHECK TO SEE IF DELTAS ARE TOO CLOSE *****
                IF((deltal-deltau).lt.1E-5) then
                type 3
3               FORMAT(' DELTA CAN NO LONGER BE signif. INCREASED.')
                RETURN
                END IF
c             -------------------------------------------
        ELSE
c       ------CRISIS ROUTE ---------------------------------------
                IT=IT+1
                IF(IT.gt.9)stop
        END IF
c       ----------------------------------------
40         CONTINUE
        WRITE(5,1)
1       FORMAT(' NO SOLUTION FOUND AFTER 100 times thru part 2')
        BOUND=.01
        RETURN
c       ----------- unsuccessful exit ----------------------------
           END
C**********************  END OF POSDEL  ***********************
c------------------------------------------------------------------------
c
c
c+++++++++++++++++++++++++ CONMAP +++++++++++++++++++++++++++++++++++++
c
c       THIS COMPUTES THE CONTRACTIVE MAP
C       OF EQN. 2.1 IN MYHRE-SAUNDERS PAPER.
C       THIS IS ONLY GOOD FOR PSEUDO-SERIES SYSTEMS
C                                   JBL 10/31/78
C               UPDATED TO JUST KEEP A(J) BETWEEN THE
```

```
C                 BOUNS ON PDELTA.      JBL    2/16/80
C*********************** CONMAP *****************************

        SUBROUTINE CONMAP(FAIL,UPBD,LWBD,GAP)
        INCLUDE 'lrcomp.FOR'
        INTEGER ORDER
        LOGICAL FAIL
        DIMENSION HOLD(100),A(100),UPBD(100)
        REAL LWBD(100)
        FAIL=.FALSE.
        TOL=.0005
C       ****  IF GAP IS VERY SMALL WE SIMPLY AVERAGE THE BOUNDS OF PDELTA.
        IF(GAP.LT.1E-4)THEN
        DO 999 J=1,ORDER
        PDELTA(J)=(LWBD(J)+UPBD(J))/2.0
999     CONTINUE
        RETURN
c        ------------------------------
        ELSE
C       MAIN PART OF WORK.
        DO 10 J=1,ORDER
        HOLD(J)=PDELTA(J)
        A(J)=HOLD(J)
10      CONTINUE
C*****  SOLVE THE CONTRACTIVE MAP
        DIST2=1.0
100     DO 40 ICE=1,1000
        DIST=0.
c               --- compute delaj for each component -------
                DO 30 J=1,ORDER
                IF(WINS(J).EQ.TRIALS(J))A(J)=1.
                IF (A(J).NE.1.0) THEN
                 AJP=0.0
                  DO 25 L=1,LLL
                  P1=1.0
                  DO 26 JL=1,ORDER
                  P1=P1*PDELTA(JL)**(AL(JL)*IG(L,JL))
26                CONTINUE
                  AJP=AJP+W(L)*AL(J)*IG(L,J)*P1
25                CONTINUE
                DELAJ=DELTA*AJP
C                 *** IF DELAJ>=WINS, DO NOT ASSIGN A(J), IT WILL BE NEG.
                       IF(DELAJ.LT.WINS(J)) THEN
                       A(J)=(WINS(J)-DELAJ)/(TRIALS(J)-DELAJ)
C                      *** WE MUST MAKE SURE A(J).STAYS BETWEEN THE BOUNDS
                       IF(A(J).LT.LWBD(J))A(J)=  (PDELTA(J)+LWBD(J))/2.0
                       IF(A(J).GT.UPBD(J))A(J)=(PDELTA(J)+UPBD(J))/2.0
                       DIST=DIST+ABS(A(J)-PDELTA(J))
c                      --------------------------------------------------
                       END IF
                END IF
30              CONTINUE
c       -------------------- new pdeltas are done ----------------------
                DO 70 L=1,ORDER
                PDELTA(L)=A(L)
70              CONTINUE
c       *****   IF APPROXIMATIONS HAVE CONVERGED ENOUGH, WE STOP ITERATING.
           DIST=DIST/ORDER
c       +++++++++++++++++++++++++++++++++++++++++++++++++++++++
           IF(DIST.LT.TOL) RETURN
c       +++++++++++++++++++++++++++++++++++++++++++++++++++++++
c       *****   IF DIST IS INCREASING, CONMAP FAILS AND WE GO TRY
c       *****   ANOTHER DELTA AND KEEP THE OLD PDELTA'S.
           IF(DIST.LT.DIST2) THEN
41              DIST2=DIST
```

```
                  ELSE
                  DO 50 J=1,ORDER
50                PDELTA(J)=HOLD(J)
                  FAIL=.TRUE.
                  RETURN
C                 ------------------------
                  END IF
40          CONTINUE
            END IF
            END
C**********************************************************
$
```

REFERENCES

[1] Lucke,J.B. and Myhre,J.M. "Delta-Exact Lower Confidence Bounds for Series System Reliability " IDS report, September 1979

[2] Madansky, A. "Approximate Confidence Limits for the Reliability of Series and Parallel Systems", Technometrics 7, (1965), p.495-503

[3] Myhre, J.M. and Saunders, S.C. "On Confidence Limits for the Reliability of Systems", Annals of Mathematical Statistics 39, No. 5, (1968), p. 1463-1472.

[4] "Delta-Exact Lower Confidence Bounds for Series and Pseudo-Series System Reliability" IDS report, February 1980

REFERENCES

[1] Barlow, R.E. and Proschan, F. (1975) _Statistical Theory of Reliability and Life Testing._ Holt, Reinhart and Winston, Inc. New York

[2] Lucke,J.B. and Myhre,J.M. "Delta-Exact Lower Confidence Bounds for Series System Reliability " IDS report, September 1979

[3] Madansky, A. "Approximate Confidence Limits for the Reliability of Series and Parallel Systems", _Technometrics_ 7, (1965), p.495-503

[4] Myhre, J.M. and Saunders, S.C. "On Confidence Limits for the Reliability of Systems", _Annals of Mathematical Statistics_ 39, No. 5, (1968), p. 1463-1472.

[5] "Delta-Exact Lower Confidence Bounds for Series and Pseudo-Series System Reliability" IDS report, February 1980

LMED

8